CAER: COST-AWARE EXPERT REPLICATION FOR LOAD BALANCING IN MIXTURE-OF-EXPERTS MULTIMODAL LARGE LANGUAGE MODEL TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Mixture-of-Experts (MoE) has become a key architecture for scaling multimodal large language models, yet its input-dependent routing can cause load imbalance in expert-parallel (EP) training, leading to stragglers and reduced training throughput. Expert replication is widely used to mitigate such imbalance, but historical load estimates may fail to track rapid workload shifts, while fine-grained, frequent replication introduces additional costs that can offset its benefits. We propose CAER, a cost-aware expert replication method for efficient multimodal MoE (MMoE) training. CAER selectively triggers rebalancing through an adaptive imbalance gate and uses a cost-aware greedy scheduler to identify minimum-cost replication plans under a calibrated cost model that jointly captures computation, communication, and replication overhead. It then dynamically replicates overloaded experts onto underloaded devices while preserving the original routing patterns. Across diverse MMoE architectures and datasets, CAER consistently outperforms state-of-the-art methods, improving end-to-end training throughput by up to 31.8% over EPLB and 31.0% over LPLB while maintaining comparable or better load balance. These results demonstrate the effectiveness and robustness of CAER across diverse training settings.

1 INTRODUCTION

The Mixture-of-Experts (MoE) architecture (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2022) has emerged as a crucial framework for efficiently scaling multimodal large language models (MLLMs) (Huang et al., 2026). By leveraging conditional computation, MoE models expand model capacity with a modest increase in per-token computational cost, as each token is routed to a small subset of specialized experts (Cai et al., 2025; Jiang et al., 2024). This expert specialization is well suited to multimodal learning (Bo et al., 2026), where inputs from different modalities exhibit distinct semantic characteristics and computational patterns that can be captured by specialized experts. This synergy is reflected in recent multimodal MoE (MMoE) models, such as Qwen3-VL-MoE (Bai et al., 2025), Kimi-VL (Team et al., 2025), and DeepSeek-VL2 (Wu et al., 2024b).

Despite their scalability, MoE models suffer from load imbalance during training due to their dynamic and input-dependent routing (Shazeer et al., 2017; Fedus et al., 2022). Such imbalance affects both model quality and system efficiency. From the optimization perspective, highly skewed routing may cause a small subset of experts to receive large numbers of tokens while others are rarely activated, potentially leading to expert under-utilization or even expert collapse (Wang et al., 2024). From a systems perspective, expert parallelism (EP) distributes experts across multiple GPU ranks. Since the execution time of an MoE layer is typically bounded by the most heavily loaded rank, overloaded GPUs become stragglers, leaving underloaded GPUs idle while waiting for synchronization (He et al., 2022; Wang et al., 2023). Such straggler effects can degrade hardware utilization and reduce training throughput (Zhang et al., 2026).

Expert replication is a widely used approach to mitigating load imbalance, where heavily loaded experts are replicated and redistributed across devices based on estimated loads (Wang et al., 2026). However, load estimates derived from historical statistics may fail to track rapid workload shifts, limiting their effectiveness under dynamic routing patterns (Jin et al., 2026; Wei et al., 2026). More

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

adaptive methods adjust expert replicas at finer granularity, even on a per-iteration or per-microbatch basis (Skiadopoulos et al., 2025; Wei et al., 2026). Such adaptation, however, introduces additional planning, communication, and expert-state movement costs that should be carefully controlled to preserve throughput gains.

Recent work has therefore begun to consider the cost-effectiveness of expert replication (Zhao et al., 2026; Zhang et al., 2026), primarily focusing on memory efficiency in MoE serving or communication overhead during training. Yet, effective load balancing for MoE training requires a more comprehensive cost model that jointly captures computation, communication, and replication overhead. Beyond assessing replication costs, it is also important to determine when rebalancing is worthwhile. Since load imbalance varies substantially across layers (He et al., 2026; Shahout et al., 2025), rebalancing should be selectively applied where its expected benefit outweighs the associated overhead, rather than indiscriminately across layers (Zhao et al., 2026).

Motivated by these observations, we propose CAER (Cost-Aware Expert Replication), a dynamic load-balancing method for multimodal MoE training. CAER, as shown in Figure 1, incorporates two key designs. The **adaptive imbalance gate** selectively triggers rebalancing using a layer-specific threshold that is adjusted via the additive-increase multiplicative-decrease (AIMD) feedback based on the estimated benefit and planning overhead. For each activated layer, a **cost-aware greedy scheduler** explores candidate replication plans and selects the minimum-cost configuration under a calibrated model that jointly captures computation, communication, and replication overhead. CAER then temporarily replicates overloaded experts onto underloaded devices and redirects selected assignments for execution, while preserving the original expert assignments and consolidating replica gradients at their canonical owners.

We evaluate CAER on MMoE models with different architectures across seven datasets spanning multimodal and language-only tasks. CAER consistently outperforms state-of-the-art methods, improving end-to-end training throughput by up to 31.8% over EPLB (DeepSeek-AI, 2025a) and 31.0% over LPLB (DeepSeek-AI, 2025b) while maintaining comparable or better load balance.

The contributions of this paper are summarized as follows:

- We propose CAER, a cost-aware expert-replication method for MMoE training. CAER combines an adaptive imbalance gate with a cost-aware scheduler that jointly models computation, communication, and replication overhead to perform selective and cost-effective load balancing.
- We conduct extensive experiments across three MMoE models and different modality datasets. CAER consistently improves end-to-end training throughput over the competitive methods while achieving comparable or better load balance. These results demonstrate the effectiveness and generalization of CAER across diverse modalities and model architectures.

2 Related Work

2.1 Multi-modal Mixture-of-Experts Models

Sparse MoE activates a small subset of experts per token, expanding model capacity without proportionally increasing computation (Shazeer et al., 2017; Lepikhin et al., 2020). Such conditional computation enables specialized processing of heterogeneous multimodal inputs through different experts (Mustafa et al., 2022). These advantages have motivated the development of a growing number of MMoE models. MMoE models can be categorized into three types according to the explicit modality conditioning in routing and expert allocation. Modality-agnostic framework models use a shared router and expert pool to process different modalities in a common representation space, such as LIMoE (Mustafa et al., 2022), MoE-LLaVA (Lin et al., 2026), Qwen3-VL (Bai et al., 2025), and DeepSeek-VL2 (Wu et al., 2024b). Modality-aware routing models, exemplified by Ming-Omni (AI et al., 2025), employ modality-specific routers over a common expert pool. Modality-partitioned framework models combine modality-specific routers with dedicated expert subsets. ERNIE-4.5-VL (Baidu-ERNIE-Team, 2025) follows this design for its routed experts, while retaining shared experts to facilitate cross-modal integration. These designs offer different balances between cross-modal parameter sharing and modality-specific processing.

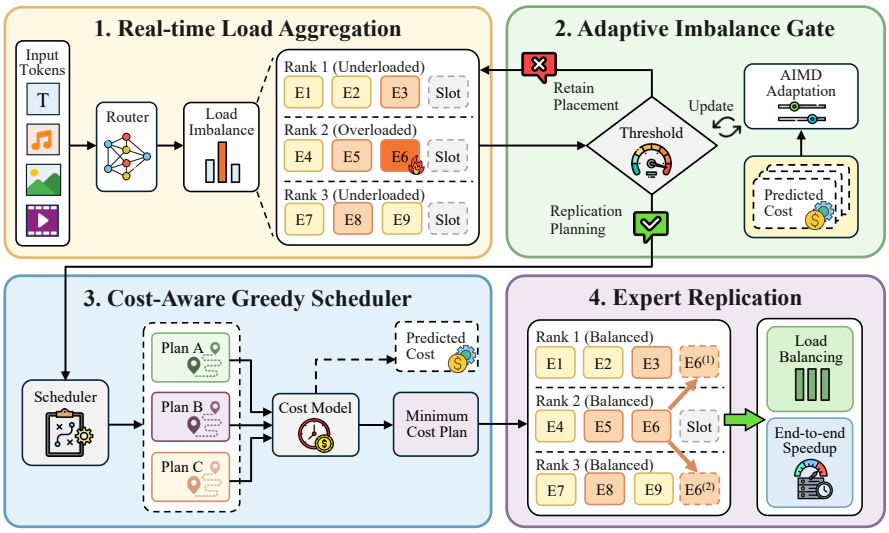


Figure 1: Overview of CAER. CAER performs selective and cost-aware expert replication for multimodal MoE training. It first aggregates real-time expert loads and uses an adaptive imbalance gate to determine whether a layer should be rebalanced, with a layer-specific threshold adjusted via AIMD feedback based on the estimated benefit and planning overhead. For each activated layer, a cost-aware greedy scheduler evaluates candidate replication plans using a calibrated cost model that jointly captures computation, communication, and replication overhead, and selects the minimum-cost plan. CAER then temporarily replicates overloaded experts onto underloaded devices to improve load balance and end-to-end training throughput while preserving the original expert assignments.

2.2 MoE Load Balancing

The input-dependent routing and sparse activation make load imbalance inherent in distributed MoE training under EP. Load imbalance can hinder model performance and also exacerbate system bottlenecks (Wang et al., 2024; He et al., 2022). Existing works mainly address load imbalance from either the algorithm or systems perspective (Cai et al., 2025). Algorithm-based approaches encourage more uniform load distribution through auxiliary loss, capacity factors, routing optimization, and loss-free bias (Lepikhin et al., 2020; Qiu et al., 2025; Fedus et al., 2022; Zhou et al., 2022; Wang et al., 2024). However, they can change the routing selection and face the trade-off between model expressiveness and load balancing (Zhang et al., 2026). Instead, system-based approaches preserve the original routing patterns and mitigate imbalance through expert placement, expert replication, or token redistribution (Nie et al., 2023; Wang et al., 2023; He et al., 2022; Wu et al., 2024a; Liu et al., 2026). Despite their effectiveness in mitigating load imbalance, expert replication methods may struggle to track rapid workload shifts when relying on historical load statistics (DeepSeek-AI, 2025a; Wang et al., 2026; Wei et al., 2026). Moreover, frequent replication can introduce non-negligible planning, communication, and migration overhead (Zhang et al., 2026; Zhao et al., 2026). To address these limitations, we propose CAER, a cost-aware expert replication approach that dynamically adapts to workload changes while accounting for the associated system costs.

3 Methodology

Figure 1 illustrates the overall workflow of CAER. Our method consists of two key components: an adaptive imbalance gate that determines whether rebalancing should be triggered, and a cost-aware greedy scheduler that determines how to rebalance by selecting a minimum-cost replication plan. In the following sections, we first formulate the load-balancing problem and its optimization objective, then introduce these two components, and finally describe the end-to-end CAER workflow.

3.1 Problem Formulation

We consider an MoE layer with E experts distributed across R EP ranks. For each valid token i, the router selects a sparse expert set $\mathcal{S}_i$, where $|\mathcal{S}_i| = K$. Under static placement P_0, expert e is assigned to its canonical owner $o(e) \in \{0, \ldots, R-1\}$. Let L_r denote the number of routed token–expert assignments processed by rank r. We quantify EP load imbalance as

$$I = \frac{\max_r L_r}{\frac{1}{R}\sum_r L_r}, \tag{1}$$

where $I = 1$ indicates perfect balance. Since EP execution is synchronous, layer latency is determined by the slowest rank:

$$T = \max_{0 \le r < R} T_r, \tag{2}$$

where T_r is the execution time of rank r.

Given fixed router decisions, we seek a feasible load-balancing strategy that changes physical execution placement while preserving the logical token–expert assignments. The objective is

$$\min_{\pi \in \Pi} \left[\max_r T_r(\pi) + T_{\mathrm{LB}}(\pi) \right], \tag{3}$$

where Π is the feasible strategy set, $T_r(\pi)$ is the processing time on rank r, and $T_{\mathrm{LB}}(\pi)$ is the exposed load-balancing overhead.

3.2 Adaptive Imbalance Gating

The benefit of load balancing can vary across layers and microbatches because their routed workloads and execution shapes differ. We therefore introduce the adaptive imbalance gating mechanism, inspired by additive-increase multiplicative-decrease (AIMD) control (Chiu & Jain, 1989; Wang et al., 2026), to activate replica planning.

Extending Eq. 1 to layer ℓ and microbatch t, the gate is:

$$b_{\ell,t} = \mathbf{1}\{I_{\ell,t} > \tau_\ell\}, \qquad I_{\ell,t} = \frac{\max_r L_{\ell,t,r}}{\frac{1}{R}\sum_{r=0}^{R-1} L_{\ell,t,r}}, \tag{4}$$

where $L_{\ell,t,r}$ is the initial assignment load of rank r under P_0, and τ_ℓ is the threshold. If $b_{\ell,t} = 0$, no replication is committed and P_0 is retained.

The adaptive imbalance gating adapts τ_ℓ using the predicted benefit:

$$g_{\ell,t} = \widehat{C}_{\ell,t}(P_0) - \widehat{C}_{\ell,t}(P^\star_{\ell,t}) - b_{\ell,t} p_t, \tag{5}$$

where $\widehat{C}_{\ell,t}$ is the estimated execution cost, $P^\star_{\ell,t}$ is the selected plan, and p_t is the latest estimate of planning overhead. We define $P^\star_{\ell,t} = P_0$ when $b_{\ell,t} = 0$, yielding $g_{\ell,t} = 0$. If planning is admitted but P_0 remains optimal, then $g_{\ell,t} = -p_t$.

For each non-overlapping window $\mathcal{W}$ of w microbatches, CAER accumulates:

$$G_\ell = \sum_{t \in \mathcal{W}} g_{\ell,t}, \qquad B_\ell = \sum_{t \in \mathcal{W}} b_{\ell,t}. \tag{6}$$

The threshold is updated as:

$$\tau_\ell \leftarrow \begin{cases} 1 + \max\{\tau_{\min} - 1,\, \beta(\tau_\ell - 1)\}, & G_\ell > 0 \ \wedge \ B_\ell > 0, \\ \min\{\tau_{\max},\, \tau_\ell + \alpha\}, & G_\ell < 0, \\ \tau_\ell, & \text{otherwise}, \end{cases} \tag{7}$$

where $\alpha > 0$ is the additive step and $\beta \in (0, 1)$ is the multiplicative factor. All layers are initialized with $\tau_\ell = \tau_{\min}$. This feedback-driven adaptation captures trade-offs between the predicted savings from replication and its associated overheads, encouraging planning when the former outweigh the latter and suppressing it otherwise.

3.3 COST-AWARE GREEDY SCHEDULER

For a microbatch whose load imbalance exceeds the adaptive threshold, the imbalance gate admits replica planning, i.e., $b_{\ell,t} = 1$. CAER then greedily constructs a bounded sequence of candidate physical execution plans, $P_0, \ldots, P_J$, where P_0 is the original placement and each P_j extends P_{j-1} with one feasible replication action. All candidate plans in this sequence are evaluated, and the one with the lowest estimated cost is selected for execution. In the following, we describe this process in detail, omitting the layer and microbatch indices for clarity.

We define a replication action as:

$$a_j = (e_j, o_j, d_j, m_j, h_j, \mathbf{q}_j), \tag{8}$$

where expert e_j is temporarily replicated from its canonical owner $o_j = o(e_j)$ to slot h_j on a receiver rank $d_j \neq o_j$. The action redirects m_j assignments from the canonical expert to this replica. Since these assignments may originate from different source ranks, $\mathbf{q}_j = (q_{j,0}, \ldots, q_{j,R-1})$ specifies their source-wise allocation, where $q_{j,s}$ is the number taken from source rank s. A feasible plan must satisfy:

$$q_{j,s} \geq 0, \qquad \sum_s q_{j,s} = m_j, \qquad \sum_{k:e_k=e} q_{k,s} \leq N_{s,e}, \tag{9}$$

where $N_{s,e}$ is the number of assignments from rank s and routed to expert e. Each rank provides at most $S_{\max}$ replica slots, and each receiver can host at most one replica of a given expert.

At each step, CAER selects an owner o_j with the largest remaining canonical load U_r, chooses its expert e_j with the largest remaining load n_e, and places the replica on a feasible receiver d_j with the smallest current physical load L_r. Here, L_r includes assignments redirected by preceding replication actions. Using the average initial rank load $\mu = \frac{1}{R}\sum_r L_r^{(0)}$ as the balancing target, CAER determines the reassigned load as:

$$m_j = \begin{cases} \min\left\{n_{e_j},\ \mu - L_{d_j}\right\}, & L_{d_j} < \mu, \\ \min\left\{n_{e_j},\ \left\lfloor \frac{U_{o_j} - L_{d_j}}{2} \right\rfloor\right\}, & \text{otherwise.} \end{cases} \tag{10}$$

When multiple actions produce the same rank-load distribution, CAER prefers the one with lower total cross-rank traffic, followed by lower peak peer traffic. To further reduce communication, the source quotas first assign workloads originating from receiver d_j to its local replica, and then draw from other sources by remaining availability. The selected action extends the current plan as $P_j = P_{j-1} \cup \{a_j\}$. This process continues until no positive feasible transfer exists or the action budget A is exhausted.

To select among the constructed plans, CAER estimates their critical-path computation and replication-induced communication. For a plan P, let $\mathcal{G}_r(P)$ denote the canonical and replicated expert groups executed on rank r, and let $n_{r,g}(P)$ be the assignment load of group g. Since grouped matrix multiplication processes assignments in hardware-sized token tiles, the bottleneck tiled workload is:

$$M(P) = \max_r \sum_{g \in \mathcal{G}_r(P)} \left\lceil \frac{n_{r,g}(P)}{s_{\text{tile}}} \right\rceil, \tag{11}$$

where s_{tile} is the platform-specific tile size. CAER estimates the execution cost by combining the bottleneck computation, cross-node communication, and replication overhead:

$$\widehat{C}(P) = c_{\text{gmm}} \sum_{q \in \{\text{fc1},\text{fc2}\}} \left\lceil \frac{\kappa_q M(P)}{c_{\text{wave}}} \right\rceil + c_{\text{net}} X(P) + c_{\text{rep}} D(P), \tag{12}$$

where κ_q converts the tiled workload into the demand of projection q, and c_{wave} is the hardware wave capacity. The terms $X(P)$ and $D(P)$ respectively denote the maximum cross-node traffic and the maximum number of replication actions involving any rank as an owner or receiver. For single-node execution, $X(P) = 0$. The coefficients c_{gmm}, c_{net}, and c_{rep} convert the three cost proxies into estimated execution time and are calibrated on the target system. c_{net} incorporates transfer multiplicity and unit conversion, while c_{gmm} is periodically corrected using sampled critical-rank timings.

After estimating the cost of each candidate, CAER selects the minimum-cost plan, breaking ties in favor of fewer replication actions:

$$j^\star = \min\left\{ \arg\min_{0\le j\le J} \widehat{C}(P_j) \right\}, \qquad P^\star = P_{j^\star}. \tag{13}$$

Because P_0 is evaluated alongside the replicated plans, CAER selects a nonempty plan only when it strictly reduces the estimated cost of the original placement. The resulting saving, $\widehat{C}(P_0) - \widehat{C}(P^\star)$, is combined with the planning overhead in Eq. 5 and fed back to the adaptive gate.

3.4 Workflow Overview

As illustrated in Figure 1, the forward pass of CAER begins by aggregating the layer-wise workload across EP ranks. Based on this aggregated workload, an adaptive imbalance gate evaluates the rank-level load skew to determine whether replica planning should be triggered. Upon activation, a cost-aware greedy scheduler explores a bounded space of candidate replication plans, leveraging a calibrated model to identify the minimum-cost configuration. To execute this optimal plan, CAER materializes the selected experts into temporary replica slots and dynamically redirects the designated assignments. Finally, the computed representations are returned to their origin ranks and aggregated using the original routing weights.

During backpropagation, each receiver computes the parameter gradients of its replicas and returns them to the canonical owner for accumulation:

$$\nabla W_e = \nabla W_e^{\text{can}} + \sum_{j:e_j=e} \nabla W_e^{(j)}. \tag{14}$$

Parameter updates are confined to the canonical parameters W_e and their optimizer states, while replica buffers are temporary and released after use. By decoupling physical execution from logical assignments and consolidating gradients at canonical owners, CAER efficiently redistributes workloads while preserving the model's original routing and optimization semantics.

4 Experiments

4.1 Experimental Settings

Models. We evaluate our proposed method on three open-source MMoE models: Qwen3-VL-30B-A3B-Instruct (Qwen3-VL) (Bai et al., 2025), Ming-Omni (AI et al., 2025), and ERNIE-4.5-VL (Baidu-ERNIE-Team, 2025), corresponding respectively to the modality-agnostic, modality-aware routing, and modality-partitioned frameworks.

Datasets. We train the MMoE models on both multimodal and language-only datasets. Specifically, multimodal datasets include: M^3IT (Li et al., 2023), LLaVA-Instruct-665K (Liu et al., 2024), and COCO2017 (Lin et al., 2014). Language-only datasets include: GSM8K (Cobbe et al., 2021), Hellaswag (Zellers et al., 2019), ARC-C, and ARC-E (Clark et al., 2018).

Baselines. We compare our method with: (1) Baseline: the vanilla MMoE models without load balancing. (2) EPLB (DeepSeek-AI, 2025a): a method that balances expert workloads by replicating heavily loaded experts and redistributing replicas across GPUs based on historical expert loads. (3) LPLB (DeepSeek-AI, 2025b): a dynamic method that leverages linear programming to redistribute tokens among redundant experts based on real-time per-batch workload statistics.

Evaluation. To ensure a fair comparison, all methods are evaluated under identical runtime configurations and hardware environments. Unless otherwise specified, we use a consistent training and evaluation protocol across all experiments. All methods resume from the same checkpoint at step 1,000 and follow an identical data sample order for 300 training steps, with the first 100 steps used for warm-up and the subsequent 200 steps for end-to-end performance evaluation.

Implementation. The main experiments are conducted on a single node equipped with 8 NVIDIA H100 (80 GB) GPUs interconnected via NVLink. FSDP2 is applied to non-MoE modules, and expert parallelism is used for MoE layers. More detailed implementations are presented in Appendix A.

Table 1: End-to-end training performance of Baseline, EPLB, and CAER across different multimodal datasets and MMoE models. Each experiment is repeated three times, and the average results are reported. Speedup is normalized to the baseline. The best performance is **bolded**.

Method	M^{3}IT		LLaVA-Instruct-665K		COCO2017	
	Speedup ↑	Load Imb. ↓	Speedup ↑	Load Imb. ↓	Speedup↑	Load Imb.↓
QWEN3-VL						
Baseline	1.00×	1.31	1.00×	1.45	1.00×	1.32
EPLB	0.98×	1.14	1.11×	1.04	1.02×	1.06
CAER	**1.10**×	**1.05**	**1.16**×	**1.02**	**1.10**×	**1.05**
MING-OMNI						
Baseline	1.00×	1.37	1.00×	1.30	1.00×	1.24
EPLB	1.08×	1.18	1.05×	1.14	1.06×	**1.07**
CAER	**1.09**×	**1.06**	**1.12**×	**1.08**	**1.12**×	1.08
ERNIE-4.5-VL						
Baseline	1.00×	1.29	1.00×	1.32	1.00×	1.16
EPLB	1.00×	**1.13**	1.03×	**1.09**	0.99×	**1.08**
CAER	**1.06**×	1.15	**1.08**×	1.11	**1.05**×	**1.08**

4.2 MAIN RESULTS

Multimodal Scenarios. As shown in Table 1, CAER consistently improves end-to-end training throughput over the baseline and EPLB across all evaluated models and multimodal datasets. It achieves $1.05\times$–$1.16\times$ speedups while reducing load imbalance up to 29.7%. Compared with EPLB, CAER consistently improves throughput (up to 12.2%) across all evaluated datasets, while maintaining better or comparable load balance. Although EPLB improves throughput in some settings, it can underperform the baseline (e.g. Qwen3-VL with M^{3}IT at $0.98\times$). These results reveal a limitation of cost-agnostic rebalancing: better load balance does not necessarily translate into higher throughput. By incorporating expert replication costs, CAER instead prioritizes rebalancing decisions that yield tangible throughput gains.

Language-only Scenarios. To further validate the robustness of CAER, we extend our experiments to four language-only datasets. As shown in Table 2, CAER consistently delivers higher training throughput across all evaluated models and datasets, outperforming the baseline, EPLB, and LPLB in every setting. This result is consistent with the trend observed on multimodal datasets, further demonstrating the effectiveness of our cost-aware design. Notably, CAER achieves its largest throughput improvement over EPLB on HellaSwag with Ming-Omni, reaching 31.8%. CAER maintains load balance comparable to the strong load-balancing baseline LPLB in most cases, while delivering up to 31.0% higher throughput on ARC-C with Ming-Omni. These results demonstrate that CAER remains effective across diverse data modalities and workload characteristics, consistently achieving a favorable trade-off between load balance and end-to-end training throughput.

4.3 IN-DEPTH ANALYSIS

Component Ablation. We conduct an ablation study to assess the effectiveness of CAER's key components. As shown in Figure 2, the w/o cost variant removes the cost-aware selection from the scheduler, while the w/o AIMD variant disables the AIMD adaptation of the imbalance gate and uses a fixed threshold of $\tau = 1.2$ across all layers. Removing either component leads to degraded throughput compared with the full CAER design, validating the contribution of both mechanisms. Although the w/o cost variant achieves slightly better load balance than CAER, its throughput decreases by 3.5%–5.5%. This suggests that better load balance may not transfer into throughput gains, highlighting the importance of accounting for both load balance and rebalancing cost.

Convergence Efficiency. We directly fine-tune Ming-Omni and Qwen3-VL on M^{3}IT for 2,000 steps, using the first 100 steps as warm-up and measuring convergence speed over the remaining steps. We report the time-to-target-loss speedup relative to the Baseline, where values above $1.0\times$ indicate faster convergence. As shown in Figure 3, CAER consistently reaches the same target loss faster than both the Baseline and EPLB across the evaluated range. These results demonstrate that CAER improves convergence efficiency, reducing the training time required to reach the same optimization objective.

Table 2: End-to-end training performance of Baseline, EPLB, LPLB, and CAER across different language-only datasets and MMoE models. Each experiment is repeated three times, and the average results are reported. Speedup is normalized to the baseline. The best performance is **bolded**.

Method	GSM8K		Hellaswag		ARC-C		ARC-E	
	Speedup ↑	Load Imb. ↓	Speedup ↑	Load Imb. ↓	Speedup ↑	Load Imb. ↓	Speedup ↑	Load Imb. ↓
QWEN3-VL								
Baseline	1.00×	1.44	1.00×	1.39	1.00×	1.38	1.00×	1.39
EPLB	0.99×	1.04	0.98×	1.05	0.87×	1.03	0.86×	1.03
LPLB	1.03×	**1.02**	0.99×	**1.03**	0.91×	**1.02**	0.87×	**1.00**
CAER	**1.13**×	**1.02**	**1.10**×	1.04	**1.02**×	1.06	**1.01**×	1.05
MING-OMNI								
Baseline	1.00×	1.30	1.00×	1.37	1.00×	1.33	1.00×	1.35
EPLB	0.88×	**1.06**	0.85×	1.09	0.91×	1.07	0.94×	1.07
LPLB	0.88×	1.08	0.86×	**1.05**	0.84×	**1.06**	0.85×	**1.06**
CAER	**1.13**×	1.09	**1.12**×	1.07	**1.10**×	1.11	**1.08**×	1.10
ERNIE-4.5-VL								
Baseline	1.00×	1.27	1.00×	1.35	1.00×	1.35	1.00×	1.37
EPLB	0.97×	1.09	1.05×	1.09	0.89×	**1.07**	0.91×	1.07
LPLB	0.99×	**1.08**	1.06×	**1.08**	0.87×	**1.07**	0.94×	**1.05**
CAER	**1.11**×	1.11	**1.09**×	1.10	**1.12**×	1.09	**1.05**×	1.12

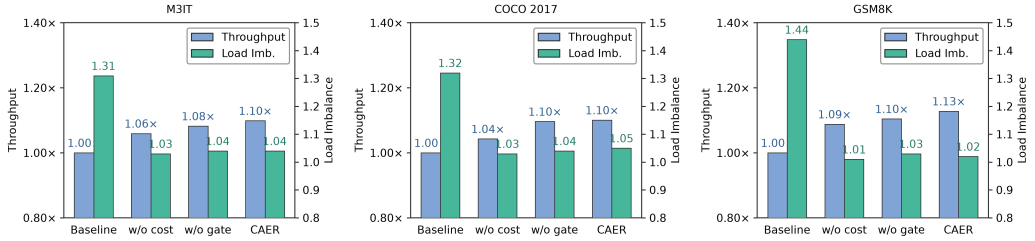


Figure 2: Ablation study of CAER on Qwen3-VL across M3IT, COCO 2017, and GSM8K. We evaluate the contributions of the cost-aware model and the adaptive AIMD control in the imbalance gate in terms of normalized throughput and load imbalance.

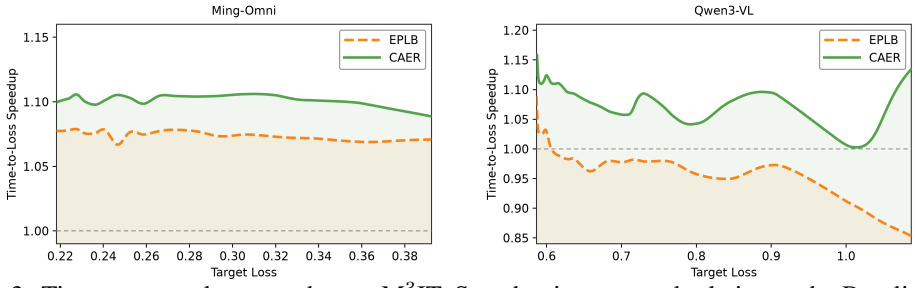


Figure 3: Time-to-target-loss speedup on M^{3}IT. Speedup is measured relative to the Baseline after a 100-step warm-up. Across target losses, CAER consistently accelerates training toward the target loss on both Ming-Omni and Qwen3-VL.

Imbalance Robustness. To evaluate robustness under varying load skew, we construct packed M^{3}IT workloads based on the baseline EP load distribution, with controlled rank imbalance ratios of approximately 1.1, 1.3, and 1.5. As shown in Figure 4, CAER consistently delivers higher throughput than EPLB while maintaining a lower EP load across different skew levels on both Qwen and Ming. Moreover, CAER's relative advantage tends to increase under higher load imbalance, demonstrating its robustness to increasing skewness.

Latency Breakdown. Figure 5 shows the per-step latency breakdown of different methods. CAER reduces the end-to-end step time by 7.9% on Ming-Omni and 8.0% on Qwen3-VL compared with the baseline. The reduction primarily comes from lower communication latency. Although CAER introduces additional load-balancing overhead, the resulting savings outweigh this cost, leading to

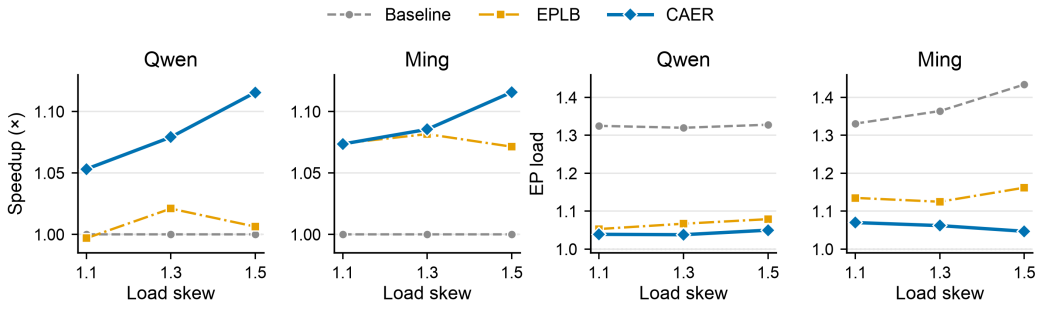


Figure 4: Sensitivity to load imbalance. The left two panels show throughput speedup over the baseline, while the right two show EP load under increasing load skew. Across both Qwen and Ming, CAER consistently achieves higher throughput and better load balance, with its advantage generally widening as load imbalance increases.

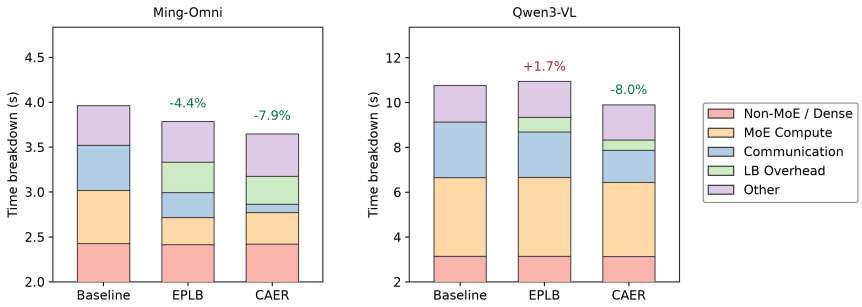


Figure 5: Training-step time breakdown on M^{3}IT across Qwen3-VL and Ming-Omni.

a reduction in training latency. In contrast, EPLB provides a smaller improvement on Ming-Omni and even increases the step time by 1.7% on Qwen3-VL.

5 Conclusion

In this work, we propose CAER, a cost-aware expert replication framework for mitigating load imbalance in multimodal MoE training. CAER selectively triggers rebalancing through an adaptive imbalance gate and identifies cost-effective replication plans using a calibrated cost model that jointly accounts for computation, communication, and replication overhead. Extensive experiments across three MMoE architectures and seven datasets demonstrate that CAER consistently outperforms both the vanilla baseline and state-of-the-art methods, achieving up to 31.8% and 31.0% higher end-to-end training throughput than EPLB and LPLB, respectively, while maintaining comparable or better load balance. These results demonstrate the importance of jointly considering load imbalance and adaptation costs for efficient MMoE training.

AI Use Statement

In this work, we used generative AI tools to assist with and polish the writing of the manuscript, including improving clarity, readability, and language. All AI-assisted text was reviewed and revised by the authors.

References

Inclusion AI, Biao Gong, Cheng Zou, Chuanyang Zheng, Chunluan Zhou, Canxiang Yan, Chunxiang Jin, Chunjie Shen, Dandan Zheng, Fudong Wang, et al. Ming-omni: A unified multimodal model for perception and generation. *arXiv preprint arXiv:2506.09344*, 2025.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Baidu-ERNIE-Team. Ernie 4.5 technical report. https://ernie.baidu.com/blog/publication/ERNIE_Technical_Report.pdf, 2025.

Zi-Hao Bo, Yaqian Li, Anzhou Hou, Rinyoichi Takezoe, Ertao Zhao, Tianxiang Pan, Jiale Yan, Mo Guang, and Kaiwen Long. Smoes: Soft modality-guided expert specialization in moe-vlms. *arXiv preprint arXiv:2604.23996*, 2026.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, 37(7):3896–3915, 2025.

Dah-Ming Chiu and Raj Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN systems*, 17(1):1–14, 1989.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

DeepSeek-AI. EPLB: Expert parallelism load balancer. https://github.com/deepseek-ai/EPLB, 2025a. GitHub repository.

DeepSeek-AI. LPLB: Linear-programming-based load balancer, 2025b. URL https://github.com/deepseek-ai/LPLB. GitHub repository.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Jiaao He, Jidong Zhai, Tiago Antunes, Haojie Wang, Fuwen Luo, Shangfeng Shi, and Qin Li. Fastermoe: modeling and optimizing training of large-scale dynamic pre-trained models. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pp. 120–134, 2022.

Shwai He, Weilin Cai, Jiayi Huang, and Ang Li. Capacity-aware inference: Mitigating the straggler effect in mixture of experts. In *International Conference on Learning Representations*, volume 2026, pp. 92274–92299, 2026.

Canbin Huang, Tianyuan Shi, Yuhua Zhu, Ruijun Chen, and Xiaojun Quan. Lookahead routing for large language models. *Advances in Neural Information Processing Systems*, 38:59042–59067, 2026.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Chao Jin, Xinming Wei, Yinmin Zhong, Chengxu Yang, Bingyang Wu, Ruidong Zhu, Zili Zhang, Yuliang Liu, and Xin Jin. Relibra: Routing-replay-guided load balancing for moe training in reinforcement learning. *arXiv preprint arXiv:2605.08639*, 2026.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

Lei Li, Yuwei Yin, Shicheng Li, Liang Chen, Peiyi Wang, Shuhuai Ren, Mukai Li, Yazheng Yang, Jingjing Xu, Xu Sun, Lingpeng Kong, and Qi Liu. M^3it: A large-scale dataset towards multi-modal multilingual instruction tuning. *CoRR*, abs/2306.04387, 2023. doi: 10.48550/ARXIV.2306.04387. URL https://doi.org/10.48550/arXiv.2306.04387.

Bin Lin, Zhenyu Tang, Yang Ye, Jinfa Huang, Junwu Zhang, Yatian Pang, Peng Jin, Munan Ning, Jiebo Luo, and Li Yuan. Moe-llava: Mixture of experts for large vision-language models. *IEEE Transactions on Multimedia*, 2026.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pp. 26286–26296. IEEE, 2024. doi: 10.1109/CVPR52733.2024.02484. URL https://doi.org/10.1109/CVPR52733.2024.02484.

Xinyi Liu, Yujie Wang, Fangcheng Fu, Xuefeng Xiao, Huixia Li, Jiashi Li, and Bin Cui. Laer-moe: Load-adaptive expert re-layout for efficient mixture-of-experts training. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pp. 1055–1072, 2026.

Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. Multimodal contrastive learning with limoe: the language-image mixture of experts. *Advances in Neural Information Processing Systems*, 35:9564–9576, 2022.

Xiaonan Nie, Xupeng Miao, Zilong Wang, Zichao Yang, Jilong Xue, Lingxiao Ma, Gang Cao, and Bin Cui. Flexmoe: Scaling large-scale sparse pre-trained model training via dynamic device placement. *Proceedings of the ACM on Management of Data*, 1(1):1–19, 2023.

Zihan Qiu, Zeyu Huang, Bo Zheng, Kaiyue Wen, Zekun Wang, Rui Men, Ivan Titov, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Demons in the detail: On implementing load balancing loss for training specialized mixture-of-expert models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5005–5018, 2025.

Rana Shahout, Colin Cai, Yilun Du, Minlan Yu, and Michael Mitzenmacher. From score distributions to balance: Plug-and-play mixture-of-experts routing. *arXiv preprint arXiv:2510.03293*, 2025.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Athinagoras Skiadopoulos, Mark Zhao, Swapnil Gandhi, Thomas Norrie, Shrijeet Mukherjee, and Christos Kozyrakis. Accelerating mixture-of-experts training with adaptive expert replication. *arXiv e-prints*, pp. arXiv–2504, 2025.

Kimi Team, Angang Du, Bohong Yin, Bowei Xing, Bowen Qu, Bowen Wang, Cheng Chen, Chenlin Zhang, Chenzhuang Du, Chu Wei, et al. Kimi-vl technical report. *arXiv preprint arXiv:2504.07491*, 2025.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-loss-free load balancing strategy for mixture-of-experts. *arXiv preprint arXiv:2408.15664*, 2024.

Wei Wang, Zhiquan Lai, Shengwei Li, Weijie Liu, Keshi Ge, Yujie Liu, Ao Shen, and Dongsheng Li. Prophet: Fine-grained load balancing for parallel training of large-scale moe models. In *2023 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 82–94. IEEE, 2023.

Yingping Wang, Yi Wu, Xiangyu Wu, Junwei Cui, Weilin Cai, Zhijiang Guo, and Jiayi Huang. Realb: Real-time load balancing for multimodal moe inference. *arXiv preprint arXiv:2604.19503*, 2026.

Xinming Wei, Chao Jin, Tuo Dai, Yinmin Zhong, Shan Yu, Chengxu Yang, Bingyang Wu, Zili Zhang, Jing Mai, Qianchao Zhu, et al. Ultraep: Unleash moe training and inference on rack-scale nodes with near-optimal load balancing. *arXiv preprint arXiv:2606.04101*, 2026.

Yongji Wu, Wenjie Qu, Xueshen Liu, Tianyang Tao, Yifan Qiao, Zhuang Wang, Wei Bai, Yuan Tian, Jiaheng Zhang, Z Morley Mao, et al. Lazarus: Resilient and elastic training of mixture-of-experts models. *arXiv preprint arXiv:2407.04656*, 2024a.

Zhiyu Wu, Xiaokang Chen, Zizheng Pan, Xingchao Liu, Wen Liu, Damai Dai, Huazuo Gao, Yiyang Ma, Chengyue Wu, Bingxuan Wang, et al. Deepseek-vl2: Mixture-of-experts vision-language models for advanced multimodal understanding. *arXiv preprint arXiv:2412.10302*, 2024b.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Lingyun Zhang, Henghua Zhang, Shilei Gu, Kai Mo, Shuai Han, Shiyong Li, Yanpeng Wang, and Dou Shen. Taot: Topology-aware optimal transport for dynamic expert replica placement in moe training. *arXiv preprint arXiv:2608.03676*, 2026.

Adrian Zhao, Zhenkun Cai, Zhenyu Song, Lingfan Yu, Haozheng Fan, Jun Wu, Yida Wang, and Nandita Vijaykumar. Craft: Fine-grained cost-aware expert replication for efficient mixture-of-experts serving. *Proceedings of Machine Learning and Systems*, 8:689–700, 2026.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

A Implementation Details

The software and hardware configurations used in our experiments are summarized in Table 3, while the evaluated models and their corresponding settings are provided in Table 4. For each model, we use the same training configuration across all datasets. Detailed training configurations for Qwen3-VL, Ming-Omni, and ERNIE-4.5-VL are provided in Tables 5, 6, and 7, respectively.

Table 3: Software and hardware configurations.

Component	**Configuration**
Python	3.11.10
PyTorch	2.7.1
CUDA	12.6
Triton	3.3.1
FlashAttention	2.7.4.post1
GPU	NVIDIA H100 (80 GB)
Intra-node Interconnect	NVLink (NV18) via NVSwitch
NVLink Bandwidth	26.562 GB/s per link, 18 links per GPU
Inter-node Network	400 GbE RoCE (ConnectX-7, MT4129)

B Additional Experimental Results

B.1 Robustness to Measurement

To ensure the effectiveness of the measurement protocol utilized in our experiments, we repeat the same 100-step warm-up and 200-step measurement protocol across different stages of the full 2,000-step fine-tuning trajectory on M^3IT. The results are shown in Table 8. For CAER, aggregating non-overlapping windows yields 1.08×/1.10× speedups on Qwen3-VL/Ming-Omni, while denser rolling windows yield 1.10×/1.11×, closely matching the 1.10×/1.09× reported in Table 1. We observe the same consistency for EPLB on Ming-Omni (1.07×/1.08× vs. 1.08× in Table 1). On Qwen3-VL, EPLB exhibits stronger training-stage dependence (0.94×/0.99× across the two trajectory-wide aggregations), but remains around or below parity with the baseline, consistent with its 0.98× result in Table 1. These trajectory-wide measurements preserve the method-level trends observed in our main evaluation, indicating that the reported speedups are representative across training stages rather than artifacts of a particular measurement window.

We further examine the sensitivity to the measurement-window length by varying it from 100 to 500 steps while keeping the 100-step warm-up and stride fixed. As shown in Table 9, CAER consistently achieves approximately 1.10× speedup on Qwen3-VL and 1.10–1.11× on Ming-Omni across all window lengths. Similarly, EPLB remains around 0.98–0.99× on Qwen3-VL and 1.08× on Ming-Omni. These results demonstrate that the observed trends are robust to the measurement-window length.

Table 4: Evaluated models and training parallelism settings.

Model	Layers	Experts (Top-k)	Parallelism (Train)	N_{slot}
Qwen3-VL-30B-A3B-Instruct	48	128 (8)	EP8	2
Ming-Lite-Omni-1.5	28	64 + 2 shared (4)	EP8	2
ERNIE-4.5-VL-28B-A3B	28	64 text + 64 multimodal + 2 shared (6)	EP8	2

Table 5: Training hyperparameters for Qwen3-VL.

Hyperparameter	**Setting**
Model	Qwen3-VL
Sequence length	3072
Micro batch size (per GPU)	6
Global batch size	96
Gradient accumulation steps	2
Learning rate	1×10^{-5}
Learning rate scheduler	Linear warmup + Cosine decay
Optimizer	AdamW
Weight decay	0
Precision	BF16, FP32 expert/dense masters

Table 6: Training hyperparameters for Ming-Omni.

Hyperparameter	**Setting**
Model	Ming-Omni
Sequence length	2048
Micro batch size (per GPU)	4
Global batch size	32
Gradient accumulation steps	1
Learning rate	5×10^{-6}
Learning rate scheduler	Cosine decay, no warmup
Optimizer	AdamW
Weight decay	0.01
Precision	BF16, FP32 expert masters

Table 7: Training hyperparameters for Ernie-4.5-VL.

Hyperparameter	**Setting**
Model	Ernie-4.5-VL
Sequence length	2048
Micro batch size (per GPU)	4
Global batch size	32
Gradient accumulation steps	1
Learning rate	1×10^{-5}
Learning rate scheduler	Linear warmup + Cosine decay
Optimizer	AdamW
Weight decay	0.1
Precision	BF16, FP32 router/gate

Table 8: Robustness to measurement windows on M^{3}IT. We report aggregate speedup over six non-overlapping windows and 18 rolling windows along the 2,000-step trajectory. The range denotes the min–max speedup across rolling windows.

Model	Method	Main	Non-overlap	Rolling	Range
Qwen3-VL	EPLB	0.98×	0.94×	0.99×	0.90–1.07×
	CAER	**1.10**×	**1.08**×	**1.10**×	1.00–1.21×
Ming-Omni	EPLB	1.08×	1.07×	1.08×	1.07–1.08×
	CAER	**1.09**×	**1.10**×	**1.11**×	1.09–1.12×

Table 9: Sensitivity to measurement-window length on M^{3}IT. We report aggregate speedup over rolling windows along the 2,000-step trajectory, with a fixed 100-step warm-up and a 100-step stride.

Model	Method	100 steps	200 steps	300 steps	500 steps
Qwen3-VL	EPLB	0.98×	0.99×	0.99×	0.99×
	CAER	**1.10**×	**1.10**×	**1.10**×	**1.10**×
Ming-Omni	EPLB	1.08×	1.08×	1.08×	1.08×
	CAER	**1.11**×	**1.11**×	**1.11**×	**1.10**×